OPTIMIZATION OF THE BUGS CLASIFICATION OF THE TICKETING SYSTEM IN SOFTWARE DEVELOPMENT: A STUDY CASE

Danar Ardhito[1] and Abba Suganda Girsang[2]
Master of Information Technology, Binus Graduate Programs, Bina Nusantara University
Jakarta 11480, Indonesia
Email: [1]danarardhito@gmail.com, [2]agirsang@binus.edu

***Abstract*—Computer bug elimination is an important phase in the software development process. A ticketing system is usually used to classify the identified bug type and to assign a suitable developer. This system is handled manually and error prone. This paper proposes a new bug classification method using the fast string search algorithm. The method searches the error string and compares it to the full text. The approach is deployed to the software development process at PT. Selaras Anugerah Lestari and it results in a significant reduction in the average value of the time required to handle the bugs.**

Keywords: **Fast string searching; bugs; Ticketing System**

I. INTRODUCTION

PT. Selaras Anugerah Lestari is a newly established company. Its main business is the products related to hospitality. It sales hospital care unit and supporting tools. In order to run the business smoothly, it develops a number of applications such as Credit Approval Memorandum (CAM) system, Internal Memo, IT asset system, and Financial and HR system. It uses the ticketing system to record the bugs that occur during the development process. The ticketing is also used for the bug classification and to notify the relevant developers. Currently, the classification process is performed manually and it has the potential of human errors especially in the cases where the application contains many bugs.

A series of research in the area of software bug classification has been initiated since a couple of years ago [1–14]. The first research was done by Ref. [15]. They introduced a new classification system using a supervised text categorization so-called naïve Bayes (NB). The result showed that the accuracy level was about 30%. The accuracy level is a condition where the identified bug type matches the skill sets of the assigned developer. The other research were performed by Refs. [16, 17]. They used the earlier research as the basis for their research and added a number of extra features. Theirs approach is also able to provide a list of the recommendation of the developers. The results showed that the accuracy level had increased to 57% for the Eclipse software project and 74% for the Mozilla software project.

In this paper, we propose an approach to improve the process of bug classification using Fast String Searching Algorithm. Reference [18] stated that the Boyer Moore algorithm is extremely efficient in most cases in comparison to Knuth-Morris-Pratt and Brute Force algorithm. Reference [19] compared various exact string matching algorithms for virus signature-detection and they concluded that the best algorithm for a common searching purpose is Boyer Moore. The Boyer Moore algorithm has no special memory requirements and needs no preprocessing or complex coding and thus can be surprisingly fast.

Every bug consists of two important parts: the bug's description and the application's name. The bug description expresses the detail information about the error that happens in the application. Meanwhile, the application name is the application where the bug happens. Figure 1 shows the example of the bug description and the application name. Figure 1 shows all the important part of the bug which is the bug description and application name. The information is text based; thus, they perfectly match with the capacity of the current algorithm.

Received: June 24, 2016; received in revised form: July 8, 2016; accepted: July 29, 2016; available online: August 19, 2016.

Cite this article as: Danar Ardhito and Abba Suganda Girsang, "Optimization of the Bugs Classification of the Ticketing System in Software Development", CommIT (Communication & Information Technology) Journal 10(2), 85–87, 2016.

II. RESEARCH METHOD

The idea of the proposed method is to change the manual classification process with the newly automatic one using fast search string algorithm. The fast search string algorithm is an algorithm which uses the searching string and compares it to the full text. Figure 2 shows the algorithm.

Figure 3 shows an example of the fast string search algorithm where it is used to search the bug on the application name 'CAM'. Once the error is found, an appropriate software developed can assigned accordingly.

III. RESULTS AND DISCUSSION

The results of the study are depicted in Tables I and II. The former table shows the required time for bug classification without and with fast search string classification algorithm. From the table, one can see that the algorithm generally requires shorter bug classification time.

The difference between the two classification time is statistically evaluated using the paired t-test for comparison of the mean of two samples. Table II compares a number of statistics of the two samples. One can say that on average, the algorithm is able to reduce the classification time. The Pearson correlation suggests that the reduction occurs uniformly across the variation of the bug number.

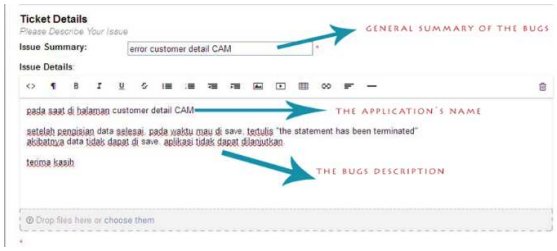

Fig. 1. The Example of the Bug's Description and Application Name.

```
      Stringlen ← Length of String
      I ← patlen
Top:  if I > stringlen then return false
      J < pathlen
Loop: if j = 0 then return j=+1
      if string(i) = pat(i) then
          J ← j - 1
          I ← i - 1
          go to Loop
      i ← i + max(delta1, (string(i)), delta2, (j))
      go to Top
```

Fig. 2. The Fast String Search Algorithm.

```
1   PAT     :  CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

2   PAT     :     CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

3   PAT     :        CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

4   PAT     :           CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

5   PAT     :              CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

6   PAT     :                 CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

7   PAT     :                    CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

8   PAT     :                       CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

9   PAT     :                          CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

20  PAT     :                                              CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL

21  PAT     :                                                 CAM
    STRING  :  PADA TAMPILAN HALAMAN FINANCIAL REVIEW APLIKASI CAM TIDAK MUNCUL
```

Fig. 3. An Example of the Process of Fast String Search Algorithm.

TABLE I
THE TEST RESULTS: BUG HANDLING TIME WITH AND WITHOUT OPTIMIZATION.

#Bugs	Bug Handling Time (MM:SS)	
	Without Optimization	With Optimization
5	00:08.40	00:06.29
10	00:16.57	00:13.42
15	00:22.59	00:20.54
20	00:29.46	00:28.51
25	00:38.26	00:34.43
30	00:47.21	00:41.06
35	00:55.57	00:49.06
40	01:04.00	00:56.23
45	01:17.27	01:01.24
50	01:24.58	01:10.13

TABLE II
THE RESULTS OF THE t-TEST FOR A COMPARISON OF TWO SAMPLES.

Statistic	Without	With
Mean	44.4	38.1
Variance	668.3	450.4
Observation	10	10
Pearson Correlation	0.99	
Hypothesized Mean Difference	0	
df	9	

Cite this article as: Danar Ardhito and Abba Suganda Girsang, "Optimization of the Bugs Classification of the Ticketing System in Software Development", CommIT (Communication & Information Technology) Journal 10(2), 85–87, 2016.

IV. CONCLUSION

Software bugs are often unavoidable in the software development process. Many software developers utilize a ticketing system for bug classifications to facilitate an efficient bug elimination process. This research proposes the use of the fast string search algorithm for the bug classification. The proposed method is evaluated empirically. The results suggest that the new approach is able to reduce the time required for the bug classification.

REFERENCES

[1] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 25–35.

[2] T. Zhang, G. Yang, B. Lee, and I. Shin, "Role analysis-based automatic bug triage algorithm," IPSJ SIG Technical Report, Tech. Rep., 2012.

[3] A. Tamrawi, T. T. Nguyen, J. M. Al-Kofahi, and T. N. Nguyen, "Fuzzy set and cache-based approach for bug triaging," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*. ACM, 2011, pp. 365–375.

[4] B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan, "Scalable statistical bug isolation," *ACM SIGPLAN Notices*, vol. 40, no. 6, pp. 15–26, 2005.

[5] S. Just, R. Premraj, and T. Zimmermann, "Towards the next generation of bug tracking systems," in *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2008, pp. 82–85.

[6] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*. ACM, 2009, pp. 111–120.

[7] P. Bhattacharya and I. Neamtiu, "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–10.

[8] D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in *2009 6th IEEE International Working Conference on Mining Software Repositories*. IEEE, 2009, pp. 131–140.

[9] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "Categorizing bugs with social networks: a case study on four open source software communities," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 1032–1041.

[10] O. Baysal, M. W. Godfrey, and R. Cohen, "A bug you like: A framework for automated assignment of bugs," in *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*. IEEE, 2009, pp. 297–298.

[11] M. M. Rahman, G. Ruhe, and T. Zimmermann, "Optimized assignment of developers for fixing bugs an initial evaluation for eclipse projects," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 439–442.

[12] Z. Lin, F. Shu, Y. Yang, C. Hu, and Q. Wang, "An empirical study on bug assignment automation using chinese bug data," in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2009, pp. 451–455.

[13] Q. Taylor, C. Giraud-Carrier, and C. D. Knutson, "Applications of data mining in software engineering," *International Journal of Data Analysis Techniques and Strategies*, vol. 2, no. 3, pp. 243–257, 2010.

[14] J. Kanwal and O. Maqbool, "Bug prioritization to facilitate bug report triage," *Journal of Computer Science and Technology*, vol. 27, no. 2, pp. 397–412, 2012.

[15] D. Čubranić, "Automatic bug triage using text categorization," in *In SEKE 2004: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*. Citeseer, 2004.

[16] J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 361–370.

[17] J. Anvik, "Automating bug report assignment," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 937–940.

[18] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[19] M. Alenezi, K. Magel, and S. Banitaan, "Efficient bug triaging using text mining," *Journal of Software*, vol. 8, no. 9, pp. 2185–2190, 2013.